Image-based virtual camera motion strategies

E. Marchand, Nicolas Courty

Proc. of Graphics Interface Conference, GI'2000
Montréal, Qc, Canada, May 2000

Image-Based Virtual Camera Motion Strategies

Eric Marchand Nicolas Courty

IRISA - INRIA Rennes
Campus de Beaulieu,
35042 Rennes Cedex, France

Abstract

This paper presents an original solution to the camera control problem in a virtual environment. Our objective is to present a general framework that allows the automatic control of a camera in a dynamic environment. The proposed method is based on the *image-based control* or visual servoing approach. It consists in positioning a camera according to the information perceived in the image. This is thus a very intuitive approach of animation. To be able to react automatically to modifications of the environment, we also considered the introduction of constraints into the control. This approach is thus adapted to highly reactive contexts (virtual reality, video games). Numerous examples dealing with classic problems in animation are considered within this framework and presented in this paper.

Key words: Automatic camera motion, Automatic cinematography, Visual servoing, Animation

1 Overview

Issues. There are numerous issues related to the control of a camera in a virtual environment. Typically, the control of the camera is handled by Lookat/lookfrom techniques associated with the definition of 3D trajectories. The camera must, usually, first position itself wrt. to its environment, and must then react in an appropriate and efficient way to modifications of the environment. As regards with the first issue, even if a full knowledge of the scene is available, as in the computer animation context, the positioning task is not a trivial problem (see [2]). There is a need for precise control of the 6 degrees of freedom (d.o.f) of the camera in the 3D space. The second issue, that can be defined as the introduction of constraints in the camera trajectory, is even more complex. In order to be able to consider unknown or dynamic environments and to achieve real-time camera motion control, these constraints must be properly modeled and "added" to the positioning task.

Related work. Visual servoing has proved, within the robotics context, to be an efficient solution to these problems. Visual servoing or image-based camera control consists in specifying a task (mainly positioning or target tracking tasks) as the regulation in the image of a set of visual features [17, 6, 8]. A good review and introduction to visual servoing can be found in [10]. As the task specification is carried out in 2D space, it does not require a 3D relationship between objects. However, since the approach is local, it is not *a priori* possible to consider planning issues. If the control law computes a motion that leads the camera to undesired configurations (such as occlusions, obstacles), visual servoing fails. Control laws taking into account these "bad" configurations therefore have to be considered. Framework that allows the consideration of such constraints has been presented in, for example, [13, 12]. It combines the regulation of the vision-based task with the minimization of cost functions reflecting the constraints imposed on the trajectory.

Viewpoints computation also has received attention in computer graphics. The main difference wrt. computer vision or robotics is that the problem is no longer ill-posed. Indeed, in that case a full knowledge of the scene is available. Even in an interactive context, the past and current behavior of all the objects is fully known. Ware and Osborn [16] consider various metaphors to describe a six d.o.f. camera control including "*eye in hand*". Within this context, the goal was usually to determine the position of the "eye" wrt. its six d.o.f in order to see an object or a set of objects at given locations on the screen. User interfaces such as a 3D mouse or a six d.o.f joystick could be considered to control such virtual device. Obtaining smooth camera motions required a skilled operator and has proved to be a difficult task. The classical lookat/lookfrom/vup parameterization is a simple way to achieve a focusing task on a world-space point. However specifying a complex visual task within the lookat/lookfrom framework is quite hopeless. Attempts to consider this kind of problem have been made by Blinn [2], however the proposed solutions appear to be dedicated to specific problems and hardly scaled to more complex tasks. Image-based control has been described within the computer graphics context by Gleicher and Witkin in [7], who called it "*Through-the-lens camera control*". They proposed to achieve very simple tasks such as positioning a camera with respect to objects defined by static "virtual" points. This technique, very similar to the visual servoing framework, considers a local inversion of the nonlinear perspective viewing transformation. A constraint optimization is used to compute the camera velocity from the desired motion of the virtual point in the image. Another formulation of the same problem has been proposed in [11]. In both case, the image Jacobian (that links the motion of the features to camera motion) is proposed only for point features. Furthermore, the introduction of constraints in the camera trajectory is not considered within the proposed framework.

The introduction of constraints has received great attention in both the robotics (e.g. [15, 4]) and computer graphics [5] communities. The resulting solutions are often similar. Each

constraint is defined mathematically as a function of the camera parameters (location and orientation) to be minimized using deterministic (e.g. gradient approaches) or stochastic (e.g. simulated annealing) optimization processes. These approaches feature numerous drawbacks. First they are usually time consuming (the search space is of dimension six) and the optimization has to be considered for each iteration of the animation process (i.e. for each new frame). It is then difficult to consider these techniques for reactive applications such as video games. As already stated, visual servoing allows the introduction of constraints in the camera trajectory [14, 13, 12]. These constraints are modeled as a cost function to be minimized. The resulting motion, also named secondary task, is then projected in the null space of the main task; it has then no effect on the main visual task. In this framework, as the camera trajectory that ensures both the task and the constraints is computed locally, it can be handled in real-time as required by the considered applications.

Presented system and contributions. The aim was to define the basic camera trajectories for virtual movie directors as well as the automatic control of a camera for reactive applications such as video games. We assume that we fully know the model of the scene at the current instant. Within this context, we present a complete framework, based on visual servoing, that allows the definition of positioning tasks wrt. to a set of "virtual visual features" located within the environment (these features can be points, lines, spheres, cylinders, etc.). When the specified task does not constrain all the camera d.o.f, the method allows the introduction of secondary tasks that can be achieved under the constraint that the visual task is itself achieved. Furthermore the considered features are not necessarily motionless. Using this approach we present solutions to various non-trivial problems in computer animation. Some of these tasks are more concerned with reactive applications (target tracking and following, obstacles and occlusion avoidance) while others deal with cinema application (panning, camera traveling, lighting conditions optimization, etc).

The remainder of this paper is organized as follows: Section 2 recalls the visual servoing framework within the task function approach. Section 3 presents methods allowing navigation in cluttered dynamic environments. Section 4 handles constraints more closely related to the cinema industry.

2 Image-based camera control

Image-based visual servoing consists in specifying a task as the regulation in the image of a set of visual features [6][8]. Embedding visual servoing in the task function approach [14] allows the use of general results helpful for the analysis and the synthesis of efficient closed loop control schemes. A good review and introduction to visual servoing can be found in [10].

2.1 Camera positioning wrt. visual targets

Let us denote $\mathbf{P}$ the set of selected visual features used in the visual servoing task measured from the image, or by projection in the computer graphics context, at each iteration of the control law. To ensure the convergence of $\mathbf{P}$ to its desired value $\mathbf{P_d}$, we need to know the interaction matrix (or image Jacobian) $\mathbf{L_P^T}$ that links the motion of the object in the image to the camera motion. It is defined by the now classic equation [6] :

$$\dot{\mathbf{P}} = \mathbf{L_P^T}(\mathbf{P}, \mathbf{p})\mathbf{T_c} \tag{1}$$

where $\dot{\mathbf{P}}$ is the time variation of $\mathbf{P}$ (the motion of $\mathbf{P}$ in the image) due to the camera motion $\mathbf{T_c}$. The parameters $\mathbf{p}$ involved in $\mathbf{L_P^T}$ represent the depth information between the considered objects and the camera frame. A vision-based task $\mathbf{e}_1$ is defined by:

$$\mathbf{e_1} = \mathbf{C}(\mathbf{P} - \mathbf{P_d}) \tag{2}$$

where $\mathbf{C}$, called combination matrix, has to be chosen such that $\mathbf{CL_P^T}$ is full rank along the desired trajectory $r \in SE^3$. If $\mathbf{e_1}$ constrains the 6 d.o.f, it can be defined as $\mathbf{C} = \mathbf{L_P^{T^+}}(\mathbf{P}, \mathbf{p})$. We will see in Section 2.3 how to define C if the 6 d.o.f are not constrained. $\mathbf{L}^+$ is the pseudo inverse of matrix $\mathbf{L}$.

To make $\mathbf{e_1}$ decreases exponentially and behaves like a first order decoupled system, the camera velocity given as input to the virtual camera is given by:

$$\mathbf{T_c} = -\lambda \mathbf{e_1} \tag{3}$$

where λ is a proportional coefficient.

Within this framework we can easily perform positioning tasks wrt. to any object of the scene. The main advantage of this approach is that even if the task is specified within the 2D image space, control is performed in 3D.

2.2 Building new skills

One of the difficulties in image-based visual servoing is to derive the image Jacobian $\mathbf{L}^T$ which corresponds to the selected control features. A systematic method has been proposed to analytically derive the interaction matrix of a set of control features defined upon geometrical primitives [6]. Any kind of visual information can be considered within the same visual servoing task (coordinates of points, line orientation, surface or more generally inertial moments, distance, etc).

Knowing these interaction matrices, the construction of elementary visual servoing tasks is straightforward. A large library of elementary skills can be proposed. The current version of our system allows to define X-to-X feature-based tasks with X = {point, line, sphere, cylinder, circle, etc.}. Using these elementary positioning skills, more complex tasks can be considered by stacking the elementary Jacobians. For example if we want to build a positioning task wrt. to a segment, defined by two points $\mathbf{P_1}$ and $\mathbf{P_2}$, the resulting interaction matrix will be defined by:

$$\mathbf{L_P^T} = \begin{bmatrix} \mathbf{L_{P_1}^T} \\ \mathbf{L_{P_2}^T} \end{bmatrix} \tag{4}$$

where $\mathbf{L_{P_i}^T}$ is defined, if $\mathbf{P_i} = (X, Y)$ and z is its depth, by (See [6] for its derivation):

$$\mathbf{L_P^T} = \begin{pmatrix} -1/z & 0 & X/z & XY & -(1+X^2) & Y \\ 0 & -1/z & Y/z & 1+Y^2 & -XY & -X \end{pmatrix} \tag{5}$$

More positioning skills can thus be simply defined.

2.3 Introducing constraints

If the vision-based task does not constrain all the n robot d.o.f, a secondary task (that usually represents a camera trajectory constraint) can be performed. $\mathbf{C}$ is now defined as $\mathbf{C} = \mathbf{CL_P^T}$ and

we obtain the following task function:

$$\mathbf{e} = \mathbf{W}^{+}\mathbf{e_1} + (\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W})\mathbf{e_2} \quad (6)$$

where

- $\mathbf{e_2}$ is a secondary task. Usually $\mathbf{e_2}$ is defined as the gradient of a cost function h_s to be minimized ($\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}}$). This cost function is minimized under the constraint that $\mathbf{e_1}$ is realized.
- $\mathbf{W}^{+}$ and $\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W}$ are two projection operators which guarantee that the camera motion due to the secondary task is compatible with the regulation of $\mathbf{P}$ to $\mathbf{P_d}$. $\mathbf{W}$ is a full rank matrix such that Ker $\mathbf{W}$ = Ker $\mathbf{L}_\mathbf{P}^T$. Thanks to the choice of matrix $\mathbf{W}$, $\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W}$ belongs to Ker $\mathbf{L_P}$, which means that the realization of ***the secondary task will have no effect on the vision-based task*** ($\mathbf{L}_\mathbf{P}^T(\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W})\mathbf{e_2} = 0$). Let us note that, if the visual task constrains all the n d.o.f of the manipulator, we have $\mathbf{W} = \mathbf{I_n}$, which leads to $\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W} = 0$. It is thus impossible in that case to consider any secondary task.

The control is now given by:

$$\mathbf{T_c} = -\lambda\mathbf{e} - (\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W})\frac{\partial \mathbf{e_2}}{\partial t} \quad (7)$$

2.4 Tracking a mobile target

A target motion generally induces tracking errors that have to be suppressed in order to always achieve the tracking task perfectly.

In that case, the motion of the target in the image can be rewritten as:

$$\dot{\mathbf{P}} = \mathbf{L}_\mathbf{P}^T\mathbf{T_c} - \mathbf{L}_\mathbf{P}^T\mathbf{T_0} \quad (8)$$

where $\mathbf{L}_\mathbf{P}^T\mathbf{T_c}$ and $\mathbf{L}_\mathbf{P}^T\mathbf{T_0}$ are respectively the contribution of the camera velocity and of the autonomous target motion to the motion of the target in the image. The new camera velocity that suppresses the tracking errors is then given by:

$$\mathbf{T_c} = -\lambda\mathbf{e} - (\mathbf{I_n} - \mathbf{W}^{+}\mathbf{W})\frac{\partial \mathbf{e_2}}{\partial t} - \alpha\mathbf{T_0} \quad (9)$$

where $\alpha \in [0, 1]$ is a scalar. If $\alpha = 1$, the tracking errors are fully suppressed while if $\alpha = 0$, they are not handled.

3 Reactive viewpoint planning

The positioning tasks that can be considered within the framework presented in the previous section are quite simple. As we did not consider the environment, the target was assumed to be "alone". We now present a method that makes it possible to achieve far more complex tasks in dynamic "*cluttered environments*". In this difficult context we will propose a purely reactive framework in order to avoid undesirable configurations in an animation context.

3.1 Avoiding obstacles

Obstacle avoidance is a good example of what can be easily given within the proposed framework. Let us assume that the camera is moving in a cluttered environment while focusing on a visual target. The goal is to ensure this task while avoiding all the obstacles in the scene.

There are in fact multiple solutions to this problem: one solution is to planify a trajectory that avoids the obstacles using a trajectory planning process. Another solution is to consider a secondary task that uses the redundant d.o.f of the camera to move away from obstacles. This function will tend to maximize the distance between the camera and the obstacle. A good cost function to achieve the goal should be maximum (infinite) when the distance between the camera and the obstacle is null. The simplest cost function is then given by:

$$h_s = \alpha\frac{1}{2\|C - O_c\|^2} \quad (10)$$

where $C(0, 0, 0)$ is the camera location and $O_c(x_c, y_c, z_c)$ are the coordinates of the closest obstacle to the camera, both expressed in the camera frame (note that any other cost function that reflects a similar behavior suits the problem). If $O_s(x_s, y_s, z_s)$ are the coordinates of the obstacle within the scene frame (or reference frame) and $M_c(RT)$ the homogenous matrix that describes the camera position within this reference frame, the obstacle coordinates within the camera frame are given by $X_c = R^T X_s - R^T T$.

The components of the secondary task are given by:

$$\mathbf{e_2} = -(x_c, y_c, z_c, 0, 0, 0)^T\frac{h_s^2}{\alpha} \quad \text{and} \quad \frac{\partial \mathbf{e_2}}{\partial t} = 0 \quad (11)$$

Multiple obstacles can be handled considering the cost function $h_s = \sum_i \alpha\frac{1}{\|C - O_{c_i}\|^2}$.

3.2 Avoiding occlusions

The goal here is to avoid the occlusion of the target due to static or moving objects (with unknown motion). The virtual camera has to perform adequate motion in order to avoid the risk of occlusion while taking into account the desired constraints between the camera and the target. There are actually many situations that may evolve in an occlusion. The first and most simple case is a moving object that crosses the camera/target line (see Figure 1.a). Two other similar cases may be encountered: in the first one (see Figure 1.b) the target moves behind another object in the scene while in the second one (see Figure 1.c) the camera follows an undesirable trajectory and is hidden behind an object.

We will now present a general image-based approach that make it possible to generate adequate camera motion automatically to avoid occlusions [12]. In a second time we will see a simple method to determine the risk of occlusion in order to weight adequately the camera response (i.e. its velocity).

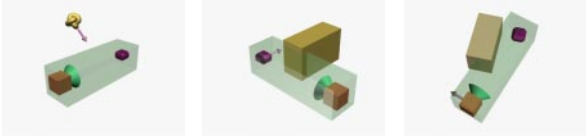

Figure 1: Occlusion issues (a) occlusion due to a moving object (b) occlusion due to the target motion (c) occlusion due to the camera motion

Automatic generation of adequate motions

Let us consider $\mathcal{O}$ the projection in the image of the set of objects in the scene which may occlude the target T: $\mathcal{O} = \{O_1, \dots O_n\}$. According to the methodology presented in paragraph 2.3 we have to define a function h_s which reaches its maximum value when the target is occluded by another object of the scene. In fact this occlusion problem can be fully defined in the image. Indeed, if the occluding object is closer than the target, when the distance between the projection of the target and the projection of the occluding object decreases, the risk of occlusion increases.

We thus define h_s as a function of this distance in the image:

$$h_s = \frac{1}{2}\alpha \sum_{i=1}^{n} e^{-\beta\left(\|T-O_i\|^2\right)} \tag{12}$$

where α and β are two scalar constants. α sets the amplitude of the control law due to the secondary task. The components of $\mathbf{e_2}$ and $\frac{\partial \mathbf{e_2}}{\partial t}$ involved in (7) are then:

$$\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}} = \frac{\partial h_s}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \mathbf{r}}, \qquad \frac{\partial \mathbf{e_2}}{\partial t} = 0$$

Computing $\frac{\partial h_s}{\partial \mathbf{P}}$ is seldom difficult. $\frac{\partial \mathbf{P}}{\partial \mathbf{r}}$ is nothing but the image Jacobian $L_{\mathbf{P}}^T$.

Let us consider the case of a single occluding object here considered as a point. The generalization to other and/or to multiple objects is straightforward. We want to see the target T at a given location in the image. Thus we will consider the coordinates $\mathbf{P} = (X, Y)$ as its center of gravity. If we also consider the occluding object $\mathcal{O}$ by a point $\mathbf{P}_\mathcal{O} = (X_\mathcal{O}, Y_\mathcal{O})$, defined as the closest point of $\mathcal{O}$ to T, we have:

$$h_s = \frac{1}{2}\alpha e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

and $\mathbf{e_2}$ is given by:

$$\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}} = \frac{\partial h_s}{\partial X}\mathbf{L}_X^T + \frac{\partial h_s}{\partial Y}\mathbf{L}_Y^T \tag{13}$$

with

$$\frac{\partial h_s}{\partial X} = -\alpha\beta(X - X_\mathcal{O})e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

and

$$\frac{\partial h_s}{\partial Y} = -\alpha\beta(Y - Y_\mathcal{O})e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

In fact $\mathbf{e_2}$ as defined in (13) is an approximation of $\frac{\partial h_s}{\partial \mathbf{r}}$. Indeed $\mathbf{L}_\mathbf{P}^T = \left[\mathbf{L}_X^T,\ \mathbf{L}_Y^T\right]^T$ is the image Jacobian related to a physical point. In our case, since the point is defined as the closest point of $\mathcal{O}$ to T, the corresponding physical point will change over time. However considering $\mathbf{L}_X^T$ and $\mathbf{L}_Y^T$ in (13) is locally a good approximation.

Risk of occlusion

Using the presented approach to compute the camera reaction is fine if the occluding object moves between the camera and the target [12] as depicted in Figure 1. Indeed, in that case occlusion will occur if no action is taken. However, it is neither necessary nor desirable to move the camera in all the cases (if the occluding object is farther than the target). A key point is therefore to detect if an occlusion may actually occur. In that case we first compute a bounding volume $\mathcal{V}$ that includes both the camera and the target at time t and at time $t + ndt$ assuming a constant target velocity (see Figure 2 and Figure 3). An occlusion will occur if an object is located within this bounding box. The time-to-occlusion may be computed as the smallest n for which the bounding box is empty. If an object $\mathcal{O}$ of the scene is in motion, in the same way, we consider the intersection of the volume $\mathcal{V}$ with a bounding volume that includes $\mathcal{O}$ at time t and at time $t + ndt$.

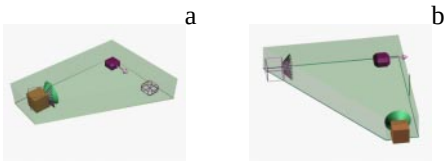

Figure 2: Computing the risk of occlusion

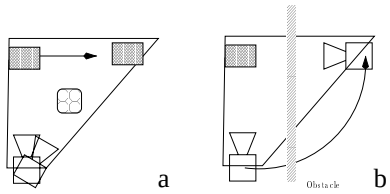

Figure 3: Detection of a future (a) occlusion (b) collision with an obstacle

Let us point out two other interesting issues:

- Obstacle avoidance may be considered in this context. Indeed, if an obstacle is on the camera trajectory, it will be located in the created bounding box (see Figure 3.b). The system will therefore forbid the camera to move in that direction.
- Some cases are more difficult to handle. A good example is a target moving in a corridor (see Figure 4). In that case, the only solution to avoid the occlusion of the target by one of the walls and to avoid the contact with the other wall is to reduce the camera/target distance. This can only be done if the z axis is not controlled by the primary task.

Figure 4: Occlusion issues: camera in a corridor

In conclusion, let us note that in this paragraph, we have just proposed a method to detect and quantify the risk of occlusion. The method proposed in paragraph 3.2 must be, in all cases,

used to generate the adequate motion that will actually avoid occlusion. The time-to-occlusion computed here will in fact be used to set the parameter α (see equation (12)) that tunes the amplitude of the response to the risk.

4 Virtual director for automatic cinematography

Whereas the issues considered in the previous section are more related to reactive applications such as video games, the problems considered in this paragraph are more concerned with camera control for movie making applications. The question considered here is the following: where should we place the camera to ensure film constraints within a given *shot* [1]. Our goal here is not to provide a director with a language that describes scenes and shots such as in [3][9] but to propose some elementary skills to be used afterwards by the director.

4.1 Cinematographic basic camera placement

Panning and Tracking Panning and tracking, certainly the most common camera motions, are straightforward to consider within the image-based framework, and have been widely considered in the previous sections of this paper. In fact the only difficulty is to choose the visual features (virtual or not) on which we want to servo. This choice is very important as it will determine the d.o.f of the virtual camera that will be used to achieve the task. For example for panning issues, the users are likely to choose one or two virtual points or a straight line as visual features (for these features the pan axes of the camera will be controlled). For tracking issues, the adequate features may depend on the desired camera motion. For example, if the camera motion has to be "parallel" to the target trajectory, the 6 d.o.f must be constrained in order to achieve a rigid link between the camera and the target (4 points or 4 lines – or any other combination of visual features such that $\mathbf{L}^T$ is a full rank 6 matrix – are then suitable for such a purpose).

Trajectory tracking As regards with the trajectory tracking issue, the problem is fairly simple. We want the camera to move on a curve $\mathcal{V}(t) = (x(t), y(t), z(t))$ defined in the camera frame. We consider a secondary task that is nothing but a function of the distance between the camera and the point $\mathcal{V}(t)$. A good solution is to define the secondary task as the function h_s simply defined as:

$$h_s = \|\mathcal{V}(t)\|^2. \tag{14}$$

Many other basic cinematographic issues exist (see [1] or [9]), e.g. building apex camera placement (that can be defined by two segments or two points for example), external or internal view (that has to consider the target and a virtual line of interest). Our goal is not to describe these tasks here. However, as they are described within the image space, image-based camera control is suitable for such issues.

4.2 Controlling lighting conditions

Controlling lighting condition (i.e. the "photography" problem), is a fundamental and non trivial issue for a film director. The main problem is to define what a good shot is wrt. these conditions.Two different functions are proposed to achieve this goal: one is directly based on the intensity within the image while the second is based on the intensity gradient (that gives information about the contrast in the image).

Our goal is to position the camera wrt. the lit aspect of the object. Therefore, we want to maximize the quantity of light (re)emitted by this object to ensure good lighting conditions. Applying the proposed methodology, we want to maximize the following cost function:

$$h_s = \frac{1}{n} \sum_X \sum_Y I(X,Y)$$

where $I(X,Y)$ represents the intensity of the 2D point (X,Y). The points (X,Y) belong to the object. The secondary task is then given by:

$$\begin{aligned} \frac{\partial h_s}{\partial \mathbf{r}} &= \frac{1}{n} \sum_X \sum_Y \left(\frac{\partial h_s}{\partial X} \frac{\partial X}{\partial \mathbf{r}} + \frac{\partial h_s}{\partial Y} \frac{\partial Y}{\partial \mathbf{r}} \right) \\ &= \frac{1}{n} \sum_X \sum_Y \left(\nabla_X \mathbf{L}_X^T + \nabla_Y \mathbf{L}_Y^T \right) \end{aligned} \tag{15}$$

where $\nabla I_X = \frac{\partial I}{\partial X}$ and $\nabla I_Y = \frac{\partial I}{\partial Y}$ represents the spatial intensity gradient.

If our goal is to maximize the contrast within the image, one possible criterion will be to maximize the sum of the spatial intensity gradient within the image. The corresponding cost function is given by:

$$h_s = \frac{1}{n} \sum_X \sum_Y \left[\nabla I_X^2 + \nabla I_Y^2 \right] \tag{16}$$

We therefore need to compute the gradient $\frac{\partial h_s}{\partial r}$ that is in fact given by

$$\frac{\partial h_s}{\partial r} = \frac{1}{n} \sum_X \sum_Y \left(\frac{\partial h_s}{\partial X} \mathbf{L}_X^T + \frac{\partial h_s}{\partial Y} \mathbf{L}_Y^T \right) \tag{17}$$

After some rewriting, we finally get:

$$\begin{aligned} \frac{\partial h_s}{\partial r} = \frac{2}{n} \sum_X \sum_Y \Bigg[& \left(\frac{\partial^2 I}{\partial X^2} \nabla I_X + \frac{\partial^2 I}{\partial Y \partial X} \nabla I_Y \right) \mathbf{L}_X^T \\ & + \left(\frac{\partial^2 I}{\partial X \partial Y} \nabla I_X + \frac{\partial^2 I}{\partial Y^2} \nabla I_Y \right) \mathbf{L}_Y^T \Bigg]^T \end{aligned} \tag{18}$$

5 Results

In this section some results are presented to illustrate our approach. Most of the images are generated in "real-time" (i.e. less than 0.1 s/frame) on a simple SUN Ultra Sparc (170Mhz) using Mesa GL (the images produced using this process can be seen in, for example, Figure 6 or Figure 8). The animations of Figure 7 or Figure 9 are computed afterwards using Maya from Alias Wavefront.

5.1 Avoiding occlusions: museum walkthrough.

In this example, we applied the proposed methodology to a navigation task in a complex environment. The target to be followed is moving in a museum-like environment. The goal is to keep the target in view (i.e. to avoid occlusions) while considering *on-line* the modifications of the environment (i.e. other moving objects). In this example, we consider a focusing task

wrt. an image centered virtual sphere. This task constrains 3 d.o.f of the virtual camera (i.e. to achieve the focusing task and to maintain the radius constant in the image). The reader can refer to [6] for the complete derivation of the image Jacobian related to a sphere. Figure 5 shows the camera trajectories for various applied strategies. Obstacles appear in yellow. The target trajectory is represented as a red dotted line, while the trajectory of another moving object is represented as a blue dotted line. The red trajectory represents the simplest strategy: just focus the object. As nothing is done to consider the environment, occlusions and then collisions with the environment occur. The blue trajectory only considers the avoidance of occlusions by static objects; as a consequence, the occlusion by the moving object occurs. The green trajectory considers the avoidance of occlusions by both static and moving objects.

A bird's eye view of some key-frames are given in Figure 6. The yellow volume (associated to the camera-target couple) corresponds to the bounding volumes used to predict the occlusions. The green volume is only used to detect the occlusions by the moving object as explained in Section 3.2. Figure 6 (A2, A3, B1) shows three views "acquired" during the avoidance of the occlusion by the *wall 1*. Between view B1 and view B2 the occlusion by the moving object is avoided. As for the first wall the problem of *wall 2* in the center of the room is handled as shown in images B3, C1 and C2. Final position is reached on C3. Figure 7 shows six views acquired by the virtual camera and rendered by using Maya.

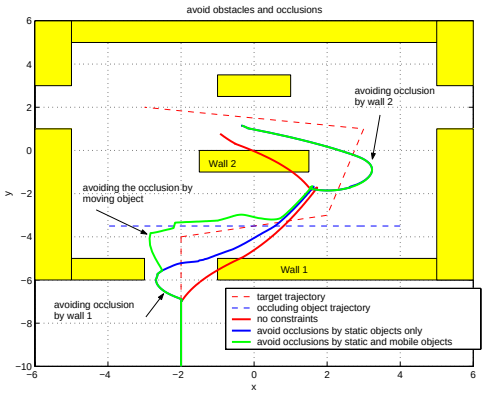

Figure 5: Museum walkthrough: camera trajectories for various strategies

5.2 Walking in a corridor: Merging multiple constraints

In this experiment the considered task is the same but the target is moving within a narrow corridor and is turning right (see Figure 8). In this experiment it is not possible to achieve this task if the distance between the camera and the target remains constant. If one wants the camera to keep the target in view an occlusion avoidance process has to be avoided. The problem is that the motion computed to avoid the occlusion moves the camera toward the red wall. An obstacle avoidance process is then necessary. We then have three secondary tasks: one related

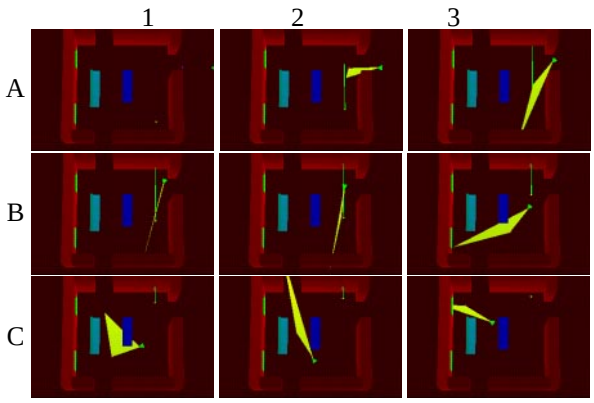

Figure 6: Museum Walkthrough: bird's eye views with the bounding volumes used for occlusion predictions

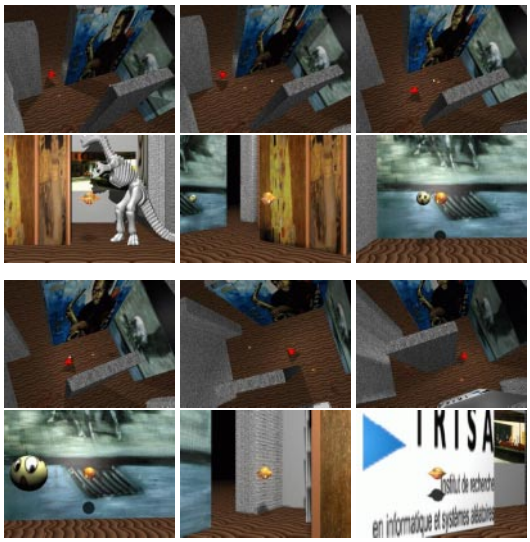

Figure 7: Museum Walkthrough: bird's eye views and corresponding camera views

to the camera-target distance, one related to obstacle avoidance (see paragraph 3.1) and the last one related to occlusion avoidance (see paragraph 3.2). The resulting control law automatically produces a motion that moves the camera away from the wall and reduces the camera-target distance. This distance, initially set to 3.5 m, decreases and reaches less that 2.5 m to ensure the task.

5.3 Trajectory tracking

In the experiment described in Figure 9, the camera focuses on the tower (i.e. we want to see this tower vertically and centered in the image). Let us note here that a similar task has been considered in [7].

Let us first consider the positioning task itself. It can be handled in various ways according to the chosen visual features. The simplest way to define a segment is to consider its two ex-

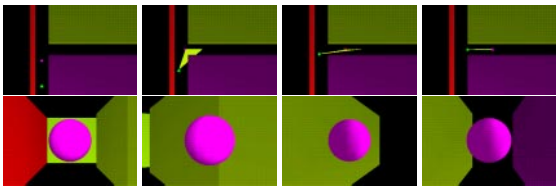

Figure 8: Moving in a corridor: bird's eye views and camera views

tremities. In that case $\mathbf{L}_{\mathbf{P}}^{T}$ is a full rank 4 matrix. In that case, the distance between the camera and the middle of the segment must remain constant. If we want to follow a trajectory that does not ensure this constraint, we will have to modify the focal length of the camera to ensure both the main task and the trajectory tracking [7]. This solution is usually not suitable for cinematographic issues. The other way to consider this segment is to choose the segment support straight line as visual feature. In that case, the image Jacobian is a full rank 2 matrix and only two d.o.f are then constrained. Figure 9.a and Figure 9.b show the beginning and the end of this focusing task. Once this is achieved, the camera follows a given 3D trajectory. Results are shown on Figure 9(b–f).

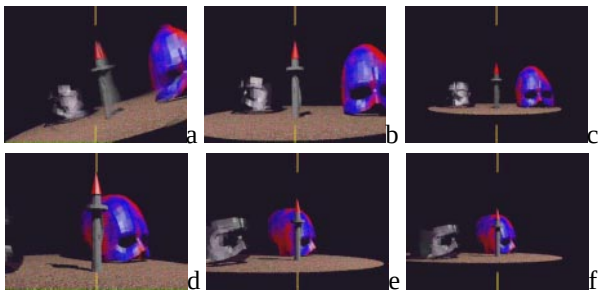

Figure 9: Positioning wrt. a segment and trajectory tracking

5.4 The "photography" problem

As regards this issue, we first perform a positioning experiment wrt. to a sphere lit by a positional light source. Results of this positioning task are presented on Figure 10(a-b). It is worth noting that the average intensity increases very smoothly (see Figure 10.c). We also plot the distance between the camera and the object-light axis (see Figure 10.d). We can note that this distance tends towards zero, i.e., the camera is located between the sphere and the light as can be predicted by theory (see Figure 10.e).

Other experiments involve more complex objects (here a teapot has been used). The results presented (see Figure 11) show the validity of our approach. Only a focusing task has been considered. This explains that the teapot turned upside down.

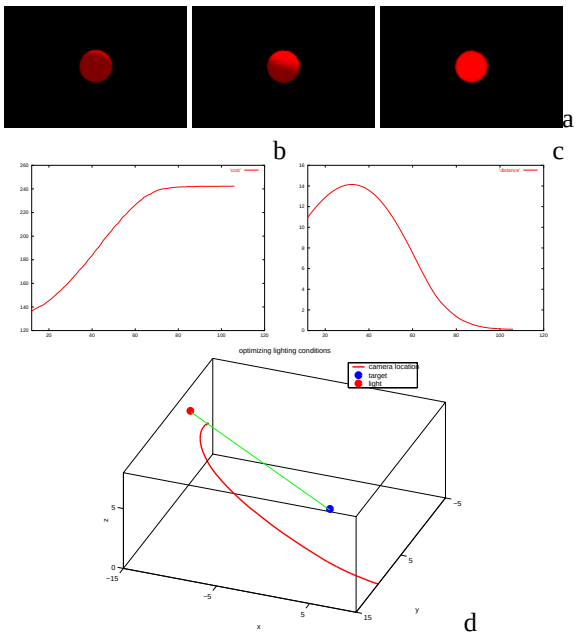

Figure 10: Positioning wrt. a sphere under good lighting conditions: (a) scene observed by the camera (illumination increases) (b) average intensity in the image (c) distance to sphere-light axis (d) camera/sphere/light position over time

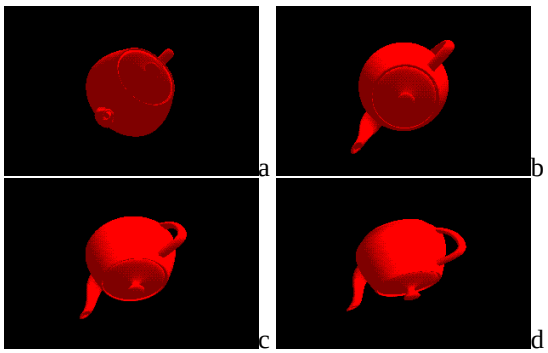

Figure 11: Teapot sequence: considering lighting conditions

6 Conclusion

There are many problems associated with the management of a camera in a virtual environment. It is not only necessary to be able to carry out a visual task (often a focusing task or more generally a positioning task) efficiently, but it is also necessary to be able to react in an appropriate and efficient way to modifications of this environment. We chose to use techniques widely considered in the robotic vision community. The basic tool that we considered is visual servoing which consists in positioning a camera according to the information perceived in the image. This image-based control constitutes the first novelty of our approach. The task is indeed specified *in a 2D space*, while the resulting camera trajectories are *in a 3D space*. It is thus a very intuitive approach of animation since it is carried out according to what one wishes to observe in the resulting images sequence.

However, this is not the only advantage of this method. Indeed, contrary to previous work [7], we did not limit ourselves to positioning tasks wrt. virtual points in static environments. In many applications (such as video games) it is indeed necessary to be able to react to modifications of the environment, of trajectories of mobile objects, etc. We thus considered the introduction of constraints into camera control. Thanks to the redundancy formalism, the secondary tasks (which reflect the constraints on the system) do not have any effect on the visual task. To show the validity of our approach, we have proposed and implemented various classic problems from simple tracking tasks to more complex tasks like occlusion or obstacle avoidance or positioning wrt. lit aspects of an object (in order to ensure good "photography"). The approach that we proposed has real qualities, and the very encouraging results obtained suggest that the use of visual control for computer animation is a promising technique. The main drawback is a direct counterpart of its principal quality: the control is carried out in the image, thus implying loss of control of the 3D camera trajectory. This 3D trajectory is computed *automatically* to ensure the visual and the secondary tasks but is not controlled by the animator. For this reason, one can undoubtedly see a wider interest in the use of these techniques within real-time reactive applications.

Acknowledgements

The authors wish to thank François Chaumette for is valuable comments and Rémi Cozot for submitting us the lighting problem.

Animations on-line.

Most of the animations presented in this paper can be found as mpeg film on the VISTA group WWW page (`http://www.irisa.fr/vista` then follow the "demo" link).

References

[1] D. Arijon. *Grammar of the Film Language*. Communication Arts Books, Hastings House, New York, 1976.

[2] J. Blinn. Where am I ? what am I looking at ? *IEEE Computer Graphics and Application*, pages 76–81, July 1998.

[3] D.B. Christianson, S.E. Anderson, L.-W. He, D.H. Salesin, D.S. Weld, and M.F. Cohen. Declarative camera control for automatic cinematography. In *Proc of AAAI'96 conference*, pages 148–155, Portland, Oregon, 1996.

[4] C.K. Cowan and P.D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE trans. on Pattern Analysis and Machine intelligence*, 10(3):407–416, May 1988.

[5] S.M. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface'94*, pages 190–199, Banff, Canada, 1994.

[6] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[7] M. Gleicher and A. Witkin. Through-the-lens camera control. In *ACM Computer Graphics, SIGGRAPH'92*, pages 331–340, Chicago, July 1992.

[8] K. Hashimoto. *Visual Servoing : Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapor, 1993.

[9] L.-W. He, M.F. Cohen, and D.H. Salesin. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *Proc. of ACM SIGGRAPH'96, in Computer Graphics Proceedings*, pages 217–224, New Orleans, August 1996.

[10] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[11] M.H. Kyung, M.-S. Kim, and S. Hong. Through-the-lens camera control with a simple jacobian matrix. In *Proc. of Graphics Interface '95*, pages 171–178, Quebec, Canada, May 1995.

[12] E. Marchand and G.-D. Hager. Dynamic sensor planning in visual servoing. In *IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 1988–1993, Lueven, Belgium, May 1998.

[13] B. Nelson and P.K. Khosla. Integrating sensor placement and visual tracking strategies. In *IEEE Int. Conf. Robotics and Automation*, volume 2, pages 1351–1356, San Diego, May 1994.

[14] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[15] K. Tarabanis, P.K. Allen, and R. Tsai. A survey of sensor planning in computer vision. *IEEE trans. on Robotics and Automation*, 11(1):86–104, February 1995.

[16] C. Ware and S. Osborn. Exploration and virtual camera control in virtual three dimensional environments. In *Proc. 90 Symposium on Interactive 3D Graphics*, pages 175–183, March 1990.

[17] L.E. Weiss, A.C. Sanderson, and C.P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, October 1987.